

Devillers Tom
Paternotte Mattéo

SAE RASPBERRY PI LAMPES, CAMÉRA ET CAPTEUR

L'objectif du projet est de réaliser une interface graphique disponible partout ou n'ou pouvons contrôler notre lampe et visionner la caméra à distance et pour finir visualiser les capteurs de température ,humidité et de luminosité .

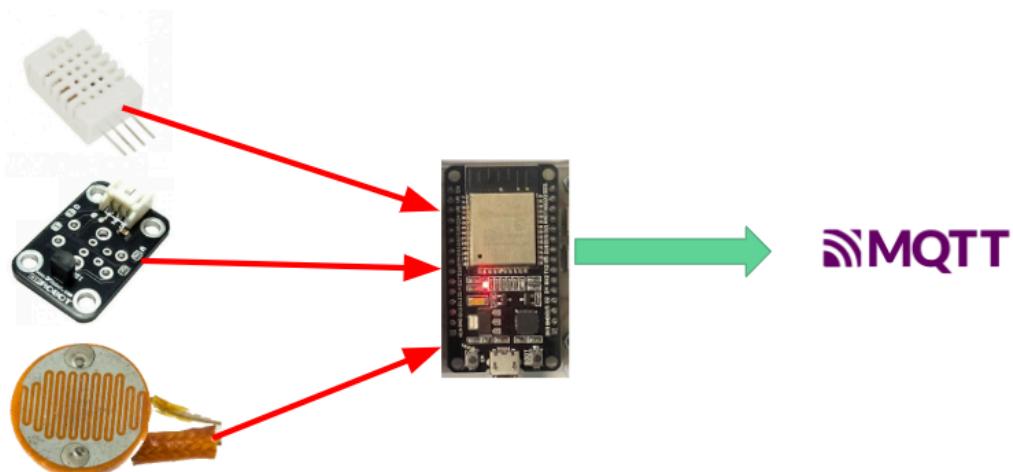
Cahier des charges :

- Faire l'acquisition des capteurs à l'aide de l'ESP32 à envoyer à la RASPBERRY PI
- Créer une interface pour recevoir des données et les stocker dans une base de données.
- Gère la lampe depuis l'interface.
- Visualiser la caméra depuis cette interfaces .
- Diffuser l'interface à l'utilisateur pouvant être disponible à tout moment en tous lieux.

Configuration des capteurs , lampes et caméra :

Pour les capteurs on va réaliser un programme à l'aide de l'esp 32 qui va devoir dans un premier temps se connecter au wifi , ensuite faire l'acquisition des capteurs de la LDR pour la luminosité , DTH22 pour l'humidité et le LM35 pour la température , et pour finire l'envoyer sur un serveur mqtt en locale :

ESP32



Voici le code arduino pour l'ESP32 :

```
#include <DHT.h>

#include "PubSubClient.h" // Connect and publish to the MQTT broker

// Code for the ESP32

#include "WiFi.h" // Enables the ESP32 to connect to the local network (via WiFi)

DHT dht(23, DHT22); //DHT PIN A6 et DHT 22 utiliser

float temperature=0;

// WiFi

const char* ssid = "toto"; // Your personal network SSID

const char* wifi_password = "motdepasse"; // Your personal network password

// MQTT

const char* mqtt_server = "192.168.215.198"; // IP of the MQTT broker

const char* humidity_topic = "humidity";

const char* temperature_topic = "temperature";

const char* ldr_topic = "ldr";

const char* mqtt_username = "12345"; // MQTT username

const char* mqtt_password = "12345"; // MQTT password

const char* clientID = ""; // MQTT client ID

// Initialise the WiFi and MQTT Client objects

WiFiClient wifiClient;

// 1883 is the listener port for the Broker
```

```
PubSubClient client(mqtt_server, 1883, wifiClient);

// Custom function to connect to the MQTT broker via WiFi

void connect_MQTT() {

    Serial.print("Connecting to ");
    Serial.println(ssid);

    // Connect to the WiFi
    WiFi.begin(ssid, wifi_password);

    // Wait until the connection has been confirmed before continuing
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    // Debugging - Output the IP Address of the ESP8266
    Serial.println("WiFi connected");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());

    // Connect to MQTT Broker
    // client.connect returns a boolean value to let us know if the connection
    // was successful.

    // If the connection is failing, make sure you are using the correct MQTT
    // Username and Password (Setup Earlier in the Instructable)
```



```
    return (temperature);
}

int lumiere() {
    int Ava3 = analogRead(35);
    int LDR=map(Ava3, 60, 4095, 1, 0);

    return(LDR);
}

// the loop routine runs over and over again forever:

void loop()
{
    delay(200);

    Serial.print("Humidite :");
    Serial.println(humidity());

    Serial.print("Temperature :");
    Serial.println(temp());

    Serial.print("ldr a :");
    Serial.println(lumiere());

    Serial.setTimeout(2000);

    int hums=humidity();
    float temps=temp();
    int ldrs=lumiere();

    delay(200);

    // PUBLISH to the MQTT Broker (topic = Humidity, defined at the beginning)

    if (client.publish(humidity_topic, String(hums).c_str())) {
```

```
Serial.println("Humidity envoyer!");

}

// Again, client.publish will return a boolean value depending on whether
it succeded or not.

// If the message failed to send, we will try again, as the connection may
have broken.

else {

    Serial.println("Humidity failed to send. Reconnecting to MQTT Broker and
trying again");

    client.connect(clientID, mqtt_username, mqtt_password);

    delay(10); // This delay ensures that client.publish doesn't clash with
the client.connect call

    if (client.publish(humidity_topic, String(hums).c_str())) {

        Serial.println("Humidity envoyer!");

    }

}

if (client.publish(temperature_topic, String(temp).c_str())) {

    Serial.println("Temperature envoyer!");

}

// Again, client.publish will return a boolean value depending on whether
it succeded or not.

// If the message failed to send, we will try again, as the connection may
have broken.

else {

    Serial.println("Temperature failed. Reconnecting to MQTT Broker and
trying again");

    client.connect(clientID, mqtt_username, mqtt_password);

    delay(10); // This delay ensures that client.publish doesn't clash with
the client.connect call

    client.publish(temperature_topic, String(temp).c_str());
}
```

```

}

    if (client.publish(ldr_topic, String(ldrs).c_str())) {

        Serial.println("ldrs envoyé!");

    }

    // Again, client.publish will return a boolean value depending on whether
    it succeeded or not.

    // If the message failed to send, we will try again, as the connection may
    have broken.

else {

    Serial.println("ldrs failed. Reconnecting to MQTT Broker and trying
again");

    client.connect(clientID, mqtt_username, mqtt_password);

    delay(10); // This delay ensures that client.publish doesn't clash with
the client.connect call

    client.publish(humidity_topic, String(ldrs).c_str());

}

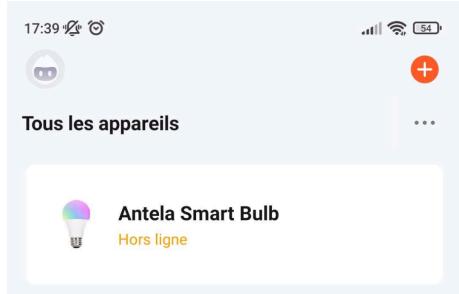
//client.disconnect(); // disconnect from the MQTT broker

delay(300*60); // print new values every 5 Minute

}

```

Ensuite nous avons réaliser la configuration de la lampes ANTULA SMART BULB avec l'application Tuya téléchargeable sur play store :



Cette configuration dans un premier temps nous permet de la connecter en wifi pour ensuite

pouvoir la contrôler à distance

Ensuite nous avons créé un compte sur Tuya IoT Platform. Après nous avons créé un projet:

The screenshot shows the 'Overview' tab of a project named 'Lampe node fin'. The project was created on 2023-11-13 16:47:45. It is categorized under 'Smart Home' and located in the 'Central Europe Data Center'. The 'Authorization Key' section displays Access ID/Client ID and Access Secret/Client Secret. A 'Cloud Authorization IP Allowlist' toggle is turned off. A note below it says: 'To improve security, you can set up a list of IPs that can legally access Tuya's data centers through the IP whitelist function. Requests from source IPs that are not in this list will be rejected.'

Ensuite nous avons lié le périphérique/devices au projet :

The screenshot shows the 'Link Tuya App Account' tab selected. It lists one account: 'matteopaternotte79@gmail.com' (UID: eu1699882547806WpU9A) which is linked to the 'Tuya Smart' project. The 'Operation' column shows a 'Manage Devices Unlink' link.

Après avoir lié le device nous avons besoin de récupérer les informations de la lampes donc nous allons dans cloud> api explorer > device management > query device detail , nous saisissons l'id que nous trouvons dans l'application tuya :

Query Device Details ⓘ

Parameter(Request Method: GET)

Params

* device_id ⓘ :

curl --request GET "https://openapi.tuya.eu.com/v2.0/cloud/thing/bfd9a8292874278730senp" --header "sign_method: HMAC-SHA256" --header "client_id: 4jv9teqeenyutrrrgw" --header "t: 1704819144192" --header "mode: cors" --header "Content-Type: application/json" --header "sign: CB098D89A2752BC568BB029645A95E77BF71C366BF85DD483CFAAA77FAF1D32" --header "access_token: 2fc6661386b52b3fb99668b5621b03d"

Request URL

Response

```
{
  "result": {
    "active_time": 1702464061,
    "bind_space_id": "172117048",
    "category": "dj",
    "create_time": 1702464061,
    "custom_name": "",
    "icon": "smart/icon/bay1603247053327grW/84ba22ecff880930081f1f39a9f77546.png",
    "id": "bfd9a8292874278730senp",
    "ip": "37.169.135.70",
    "is_online": false,
    "lat": "49.4000",
    "local_key": "4H5EG$;2.I Pv=OnC",
    "lon": "3.3300",
    "model": "A60 10W meka",
    "name": "Antela Smart Bulb",
    "product_id": "i2uk0f42o5w1xqqq",
    "product_name": "Antela Smart Bulb",
    "sub": false,
    "time_zone": "+01:00",
    "update_time": 1702902525,
    "uuid": "f82b57656b733bd9"
  },
  "success": true,
  "t": 1704819144377,
  "tid": "76824fd3af0f11ee81f17ea4c9cfb9f3"
}
```

Copy

Ensuite nous avons configuré la caméra via l'application Tapo disponible sur playstore :



Ensuite vous allez dans les paramètres de la caméra > paramètres avancés > compte de caméra ensuite crée le compte

Nom d'utilisateur :tomdevillers02 mdp : motdepasse

Ensuite vous pouvez récupérer le flux rtsp sous cette forme :

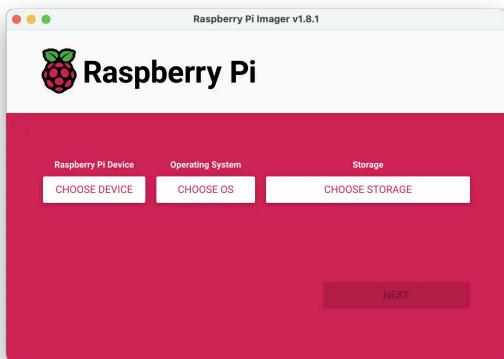
Pour un flux 1 de haute qualité : rtsp://Adresse IP/stream1

Pour un flux 2 de faible qualité : rtsp://Adresse IP/stream2

Adresse IP de la caméra disponible sur l'application

Configuration de la raspberry pi :

- On a téléchargé le OS de la raspberry sur la carte sim.



- On a ensuite téléchargé mosquitto mqtt puis on l'a paramétré.

comment installer mosquitto mqtt:

1.



`sudo apt update`

`sudo apt upgrade`

2.

`sudo apt install mosquitto mosquitto-clients`

3.

`sudo systemctl status mosquitto`

4.

`mosquitto_sub -h localhost -t "mqtt/nomtopic"`

5. Pour mettre un mdp et un nom d'utilisateur :

`sudo mosquitto_passwd -c /etc/mosquitto/passwd user`

L'identifiant sera alors stocké sous la forme user/password dans le fichier passwd

Enfin interdisez l'accès au broker par des comptes anonymes en ajoutant les lignes suivantes au fichier
/etc/mosquitto/mosquitto.conf

```
allow_anonymous false  
password_file /etc/mosquitto/passwd
```

Terminez par un redémarrage du serveur pour prendre en compte des paramètres.

```
systemctl restart mosquitto
```

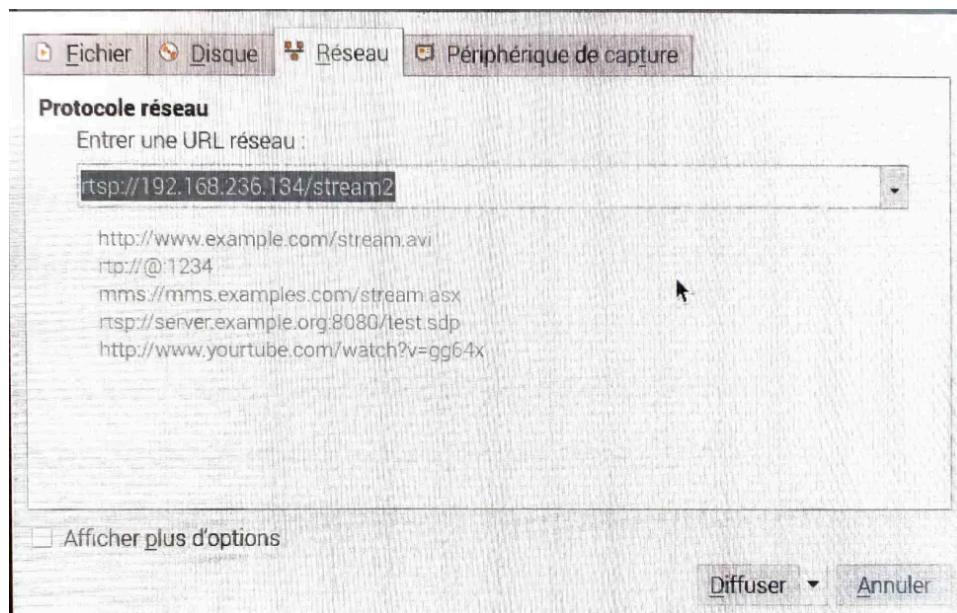


- On a téléchargé VLC pour le traitement vidéo :

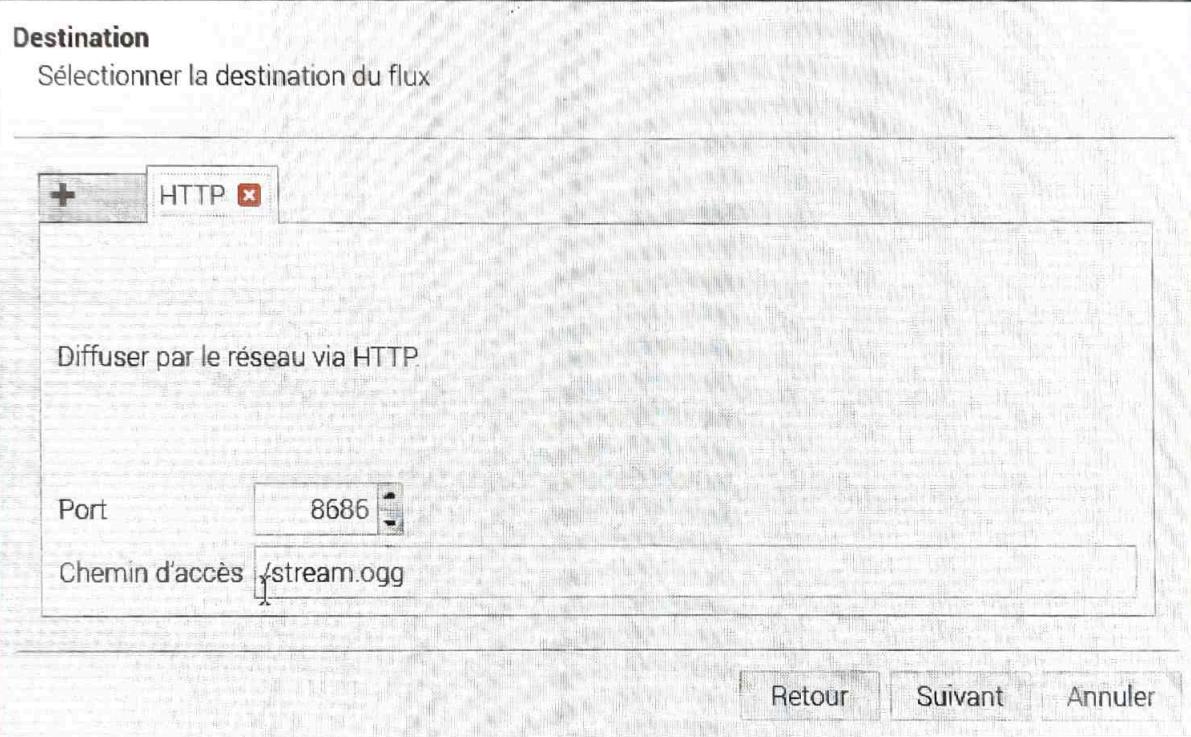
sudo apt install -y vlc

Ensuite après avoir téléchargé VLC pour convertir un flux RTSP vers un flux https

Lancer VLC ensuite aller dans DIFFUSÉE / CONVERTIRER aller dans Réseau saisir le flux RTSP://192.168.236.134/stream2 nous choisirons le stream2 pour avoir une qualité plus faible pour faciliter le traitement vidéo de la raspberry l'ip de votre caméra et la suivante :192.168.236.134



Ensuite nous avons sélectionné ceci "VIDEO - THEORA + VORBIS (OGG)"
Après ajouter une destination :



Après on à plus qu' à saisire le lien https dans le navigateur pour vérifier la bonne conversion du flux dans notre cas notre lien sera

<http://192.168.236.198:8686/stream.ogv>

L'IP sera l'adresse ip de notre raspberry pi puisque c'est là où on convertit le flux. Nous pouvons voir la caméra:

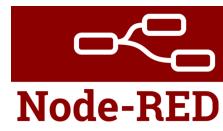
2024-01-10 09:19:46



Sources : <https://youtu.be/lqyNslhMIUA?si=ZOAmVOObYimkTTkX>

- Puis on a télécharger node-red et configurer les blocs

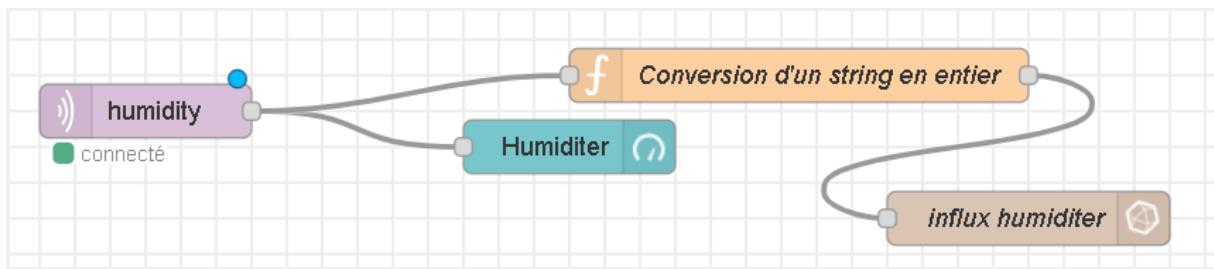
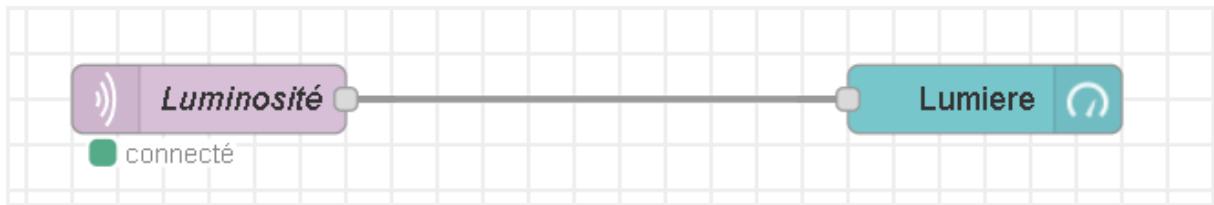
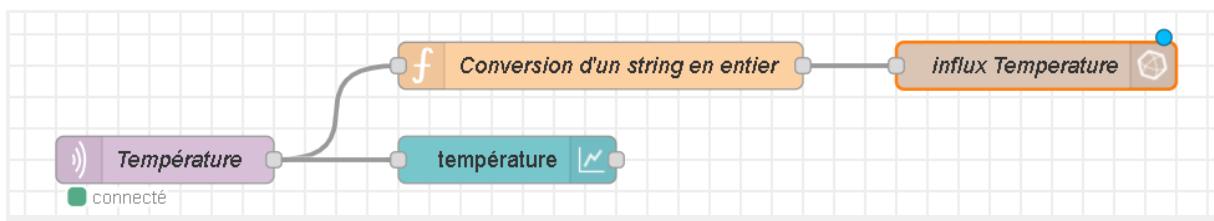
1.



```
sudo apt install build-essential
```

2.

```
bash <(curl -sL
https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```



Configuration pour le mqtt les topics doivent être pareille que dans le code arduino ainsi que dans le node red pareille pour les mdp et user :

Propriétés

Connexion **Sécurité** **Messages**

Serveur: localhost **Port**: 1883

Se connecter automatiquement
 Utiliser TLS

Protocole: MQTT V5

Client ID: Laisser vide pour s'auto générer

Restez en vie: 60

Session: Utiliser un démarrage propre
Expiration de la session (secondes):

Propriétés utilisateur: aucun

Modifier le noeud mqtt in

Supprimer

Annuler

Terminer

Propriétés



Serveur

localhost:1883



Action

S'abonner à un seul sujet



Sujet

temperature

QoS

2



Drapeaux

Ne pas recevoir les messages publiés par ce client

Conserver l'indicateur de conservation de la publication d'origine

Gestion des messages conservés

Envoyer les messages retenus



Sortie

détection automatique (objet JSON analysé, chaîne ou tampon)



Nom

Température

Modifier le noeud mqtt in

SupprimerAnnulerTerminer

Propriétés

**Serveur**

localhost:1883

**Action**

S'abonner à un seul sujet

**Sujet**

Idr

QoS

2

**Drapeaux**

- Ne pas recevoir les messages publiés par ce client
 Conserver l'indicateur de conservation de la publication d'origine

Gestion des messages conservés

Envoyer les messages retenus

**Sortie**

détection automatique (objet JSON analysé, chaîne ou tampon)

**Nom**

Luminosité

Modifier le noeud mqtt in

Supprimer Annuler Terminer

Propriétés

Serveur : localhost:1883

Action : S'abonner à un seul sujet

Sujet : humidity

QoS : 2

Drapeaux :

- Ne pas recevoir les messages publiés par ce client
- Conserver l'indicateur de conservation de la publication d'origine

Gestion des messages conservés

Envoyer les messages retenus

Sortie : détection automatique (objet JSON analysé, chaîne ou tampon)

Nom : Nom

Pour conversion une chaîne de caractère en un nombre :

Propriétés

Nom : Conversion d'un string en entier

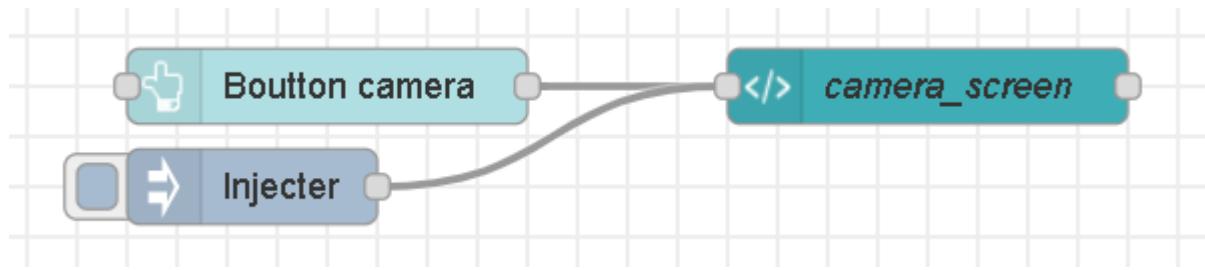
Configurations **Au démarrage** **Message reçu** **À l'arrêt**

```

1 msg.payload=Number(msg.payload);
2 return msg;

```

Caméra :



Code html pour afficher une vidéo dans un ui :

```

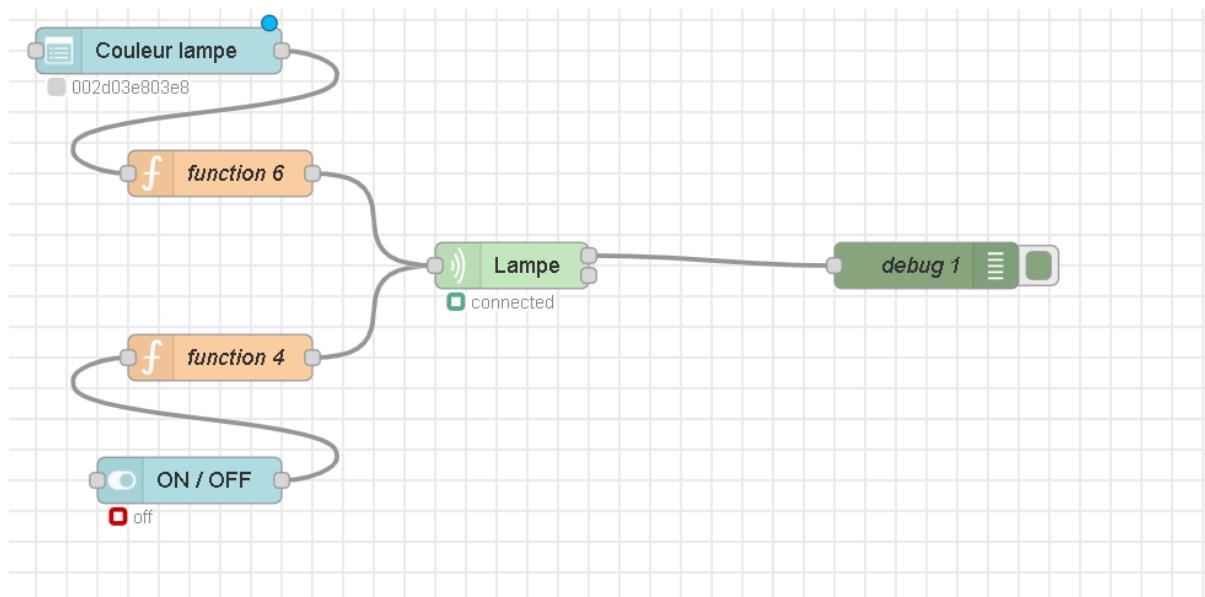
1 <div>
2   |   <video id="video" width="100%" heigth="100%"></video>
3 </div>
4
5 <script>
6   |   (function(scope) {
7   |     // Watch for messages being send to this template node
8   |     scope.$watch('msg', function (msg) {
9   |       if (msg) {
10      |         |   var video = document.getElementById('video');
11      |         |   video.src = msg.payload;
12      |         |   video.play();
13      |       }
14     });
15   })(scope);
16 </script>
17

```

Dans le boutons caméra ou dans l'injecter nous saisirons le liens https de la caméra que on à convertir avec vlc

Lampe:

Avec les informations récupère avec la création du projet sur IOT TUYA :



Configuration de la Lampe :

Device Name

Connection Details

Use Device IP or Device Virtual ID (Don't use both)

Device Virtual ID

Device Key

Store Device Id and Device Key as credentials (Credentials will not appear in the flow export)

[? How to get the device ID and Key ? Click here](#)

Device IP (Use Static IP)

Fonction 4 permet de dire à la lampes avec le dps 20 que on veut allumer ou éteindre la lampe en quelque sorte le dps indique le type d'action que on va faire à la lampe:

Modifier le noeud function

Supprimer Annuler Terminer

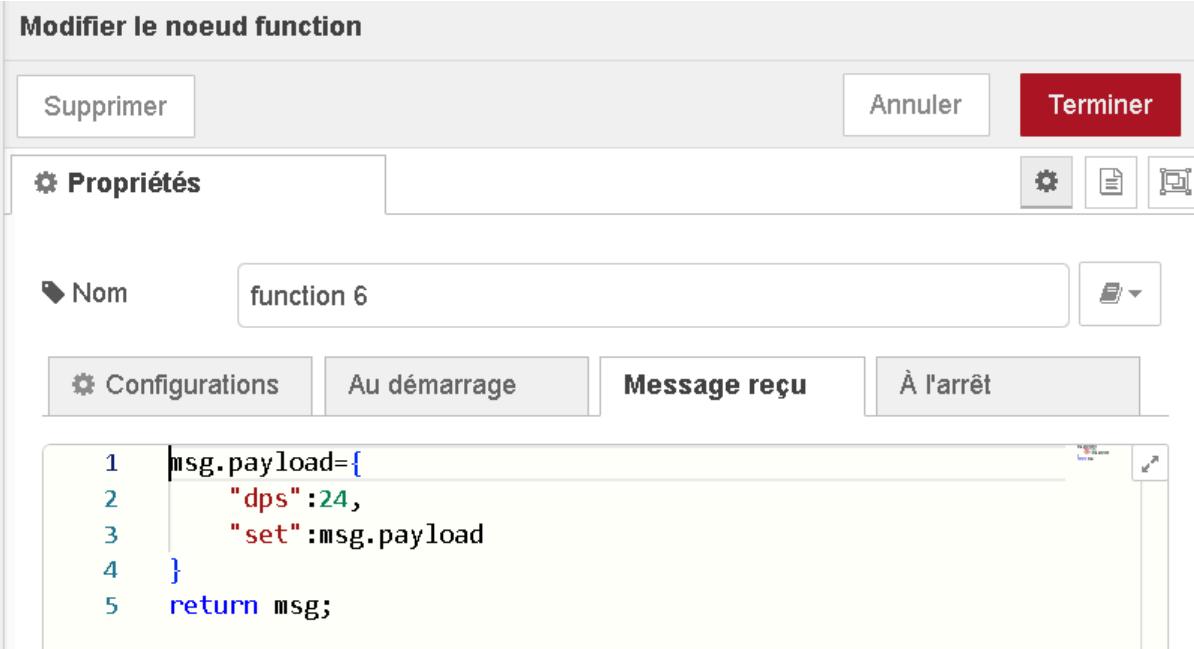
Propriétés

Nom : function 4

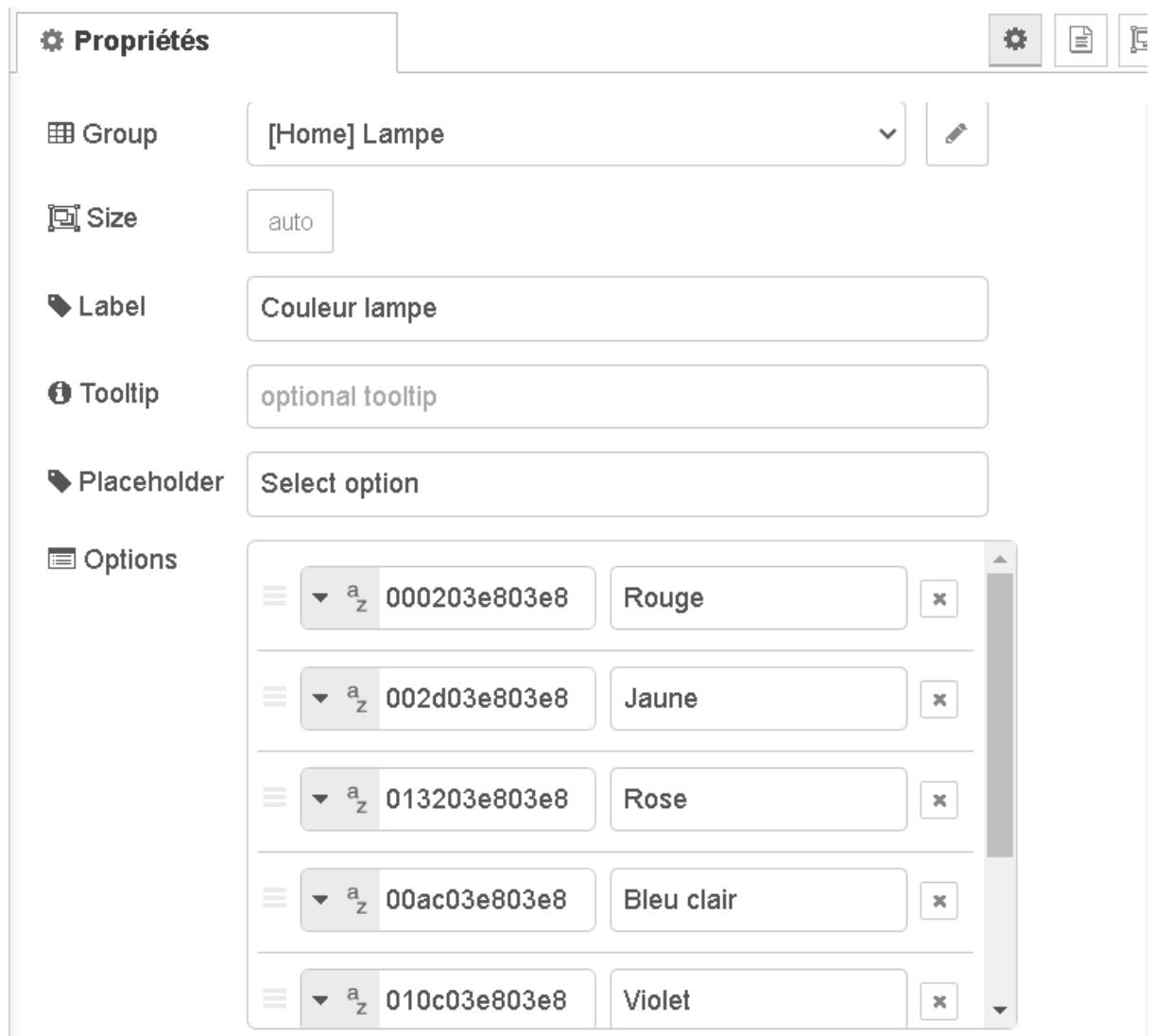
Configurations Au démarrage Message reçu À l'arrêt

```
1 msg.payload={  
2   "dps":20,  
3   "set":msg.payload  
4 }  
5 return msg;
```

Fonction 6 pour la couleur :



Pour les différentes couleur :

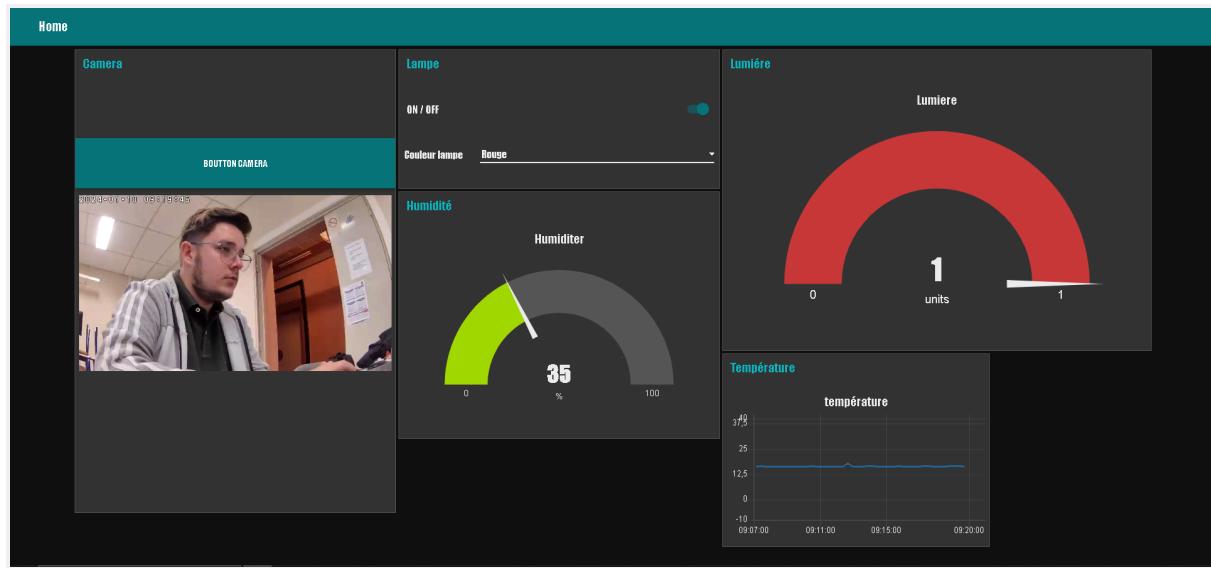


Pour déterminer quelle dps choisir il suffit de créer un événement /action à la lampes à partir de l'application de notre téléphone et de regarder sur node red le retour grâce au debug:

Par exemple on va changer la couleur et on voit le dps pour la couleur est de 24 et la donnée de la couleur est "016703e803e8"

```
10/01/2024 08:57:45 noeud: debug 1
msg.payload : Object
  ▼ object
    ▼ data: object
      ▼ dps: object
        24: "016703e803e8"
        t: 1704873461
      deviceId: "bfd9a8292874278730senp"
      deviceName: "Lampe"
```

Ensuite voici notre DASHBOARD :



- On a téléchargé influxdb pour créer des databases

1.

```
curl https://repos.influxdata.com/influxdata-archive.key | gpg  
--dearmor | sudo tee  
/usr/share/keyrings/influxdb-archive-keyring.gpg >/dev/null
```

2.

```
echo "deb  
[signed-by=/usr/share/keyrings/influxdb-archive-keyring.gpg]  
https://repos.influxdata.com/debian stable main" | sudo tee  
/etc/apt/sources.list.d/influxdb.list
```

3.

```
sudo apt update
```

4.

```
sudo apt install influxdb
```

5.

```
sudo systemctl unmask influxdb  
sudo systemctl enable influxdb
```

6.

```
sudo systemctl start influxdb
```

7.

```
influx
```

8.

```
CREATE DATABASE nomdeladatabase
```

9.

```
USE nomdeladatabase
```

10.

```
<measurement>[,<tag-key>=<tag-value>...]  
<field-key>=<field-value>[,<field2-key>=<field2-value>...]  
[unix-nano-timestamp]
```

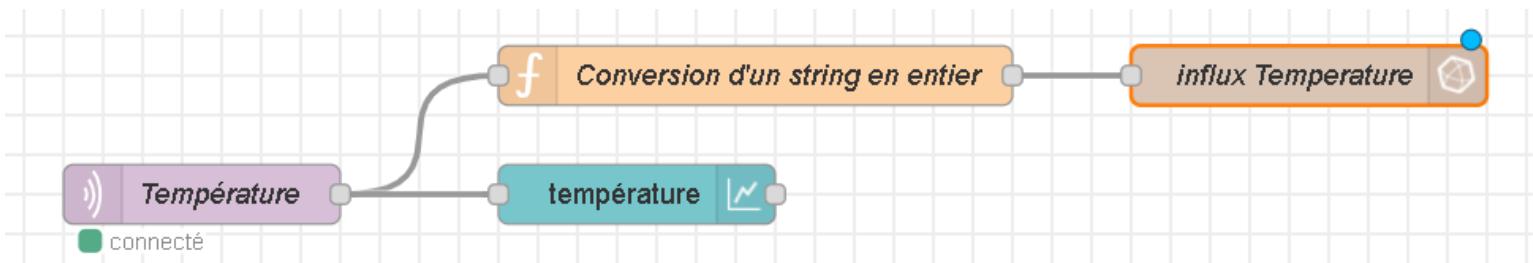
11.

Pour montrer les mesures de la database.

```
SELECT * FROM temperature
```

Puis on va rajouter sur notre node-red les block qui vont nous permettre d'envoyer les

données reçus avec le mqtt sur influxdb(les block marron).



- Pour installer Grafana :

Il suffit d'ouvrir un terminal et de taper ces commandes :

```
sudo apt update
```

```
sudo apt install grafana
```

Après quelques secondes, Grafana sera installé sur votre système. Grafana fonctionne comme un service, mais il n'est pas lancé automatiquement. Il y a une étape supplémentaire pour le démarrer, et le configurer pour qu'il démarre automatiquement au démarrage. Les commandes sont données à la fin de l'installation :

```
sudo /bin/systemctl daemon-reload
```

```
sudo /bin/systemctl enable grafana-server
```

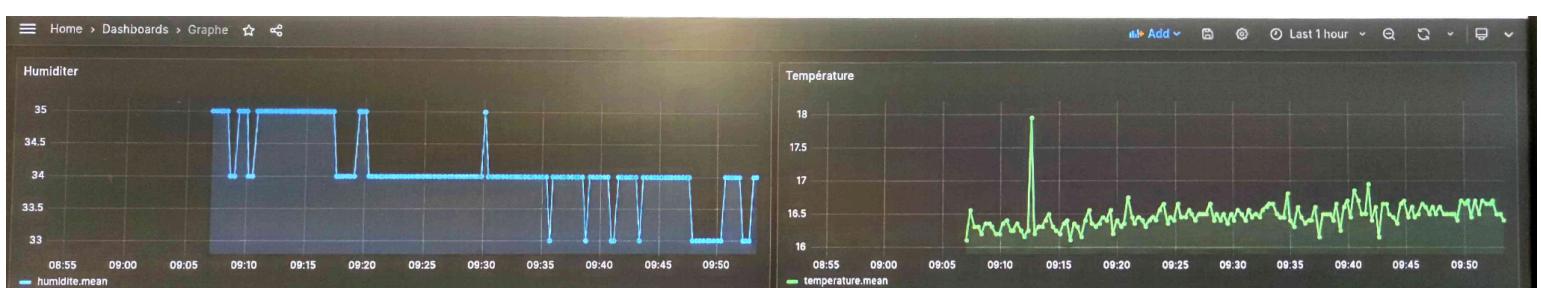
```
sudo /bin/systemctl start grafana-server
```

Ensuite on va lancer le navigateur et on saisie <http://<ip address>:3000> (IP ADRESS: de la raspberry pi) Ensuite saisir les login username : admin et password : admin .

Après avoir fait cela grafana va nous permettre de créer des graphiques(sur un dashboard). On utilise grafana car il facile d'utilisation et compatible avec influxdb.

1.On a besoin d'ajouter les database qu'on utilise.

2.Puis on crée un dashboard dans lequel on va créer plusieurs graphiques correspondant à nos databases.



- Diffuser l'interface à l'utilisateur pouvant être disponible à tout moment en tous lieux.

Nous aurons besoin de deux tunnel pour diffuser l'interface un pour la caméra et une pour le dashboard , nous utiliserons la méthode SSH TUNNELING

SSH tunneling est comme un tunnel sécurisé qui permet d'envoyer et de recevoir des données entre deux serveur . Cela rend possible l'accès à distance à des services. Dans un premier temps faire la conversions de la caméra puis lancer node-red .

Pour créer le tunnel de la caméra il faut saisir cette ligne dans le terminal:

ssh -R 80:localhost:8686 serveo.net

Ensuite on va vous retourner un lien :

<https://2365e27c54818df0308ca2a0ec087712.serveo.net/>

Ajouter à ce lien /stream.ogg

Ce qui donne :

<https://2365e27c54818df0308ca2a0ec087712.serveo.net/stream.ogg>

Saisie ce lien dans le node red dans le Boutons caméra

Ensuite réaliser le tunnel du dashboard :

Il faut saisir la commande suivante :

ssh -R 80: localhost:1880 serveo.net

Le liens qui nous à étais retourner : <https://373e12304cfa940f279820729baa15df.serveo.net>

Ajouter : /ui au lien

<https://373e12304cfa940f279820729baa15df.serveo.net/UI>

Et maintenant vous avez plus qu' à partager ce lien aux différentes personnes qui souhaite avoir accès à l'interfaces .

Conclusion :

Ce projet nous aura permis d'étudier la carte Raspberry pi 4B . Il nous aura permis d'apprendre de nous mêmes et rechercher les système qu'on peut utiliser afin de satisfaire le cahier des charges . Il nous appris la rigueur et la discipline que doit imposer un projet de cette envergure.